CASE tool for embedded systems

Lintalo Suehiro

Kongens Lyngby 2008
IMM-BEng-2008-9

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-BEng: ISSN

Abstract

In the course on "Software Engineering 2" in autumn 2007, different groups developed different parts of a simple CASE tool for modeling, simulating, and visualising the behavior of embedded systems. In this bachelor thesis, different parts of the tool which the groups developed was integrated into a single Tool. Moreover, the features for modeling the behavior of the components of a system is improved in this project.

Preface

This bachelor thesis was written at the Computer Science and Engineering section, Institute of Informatics and Mathematical Modeling at the Technical University of Denmark from February 4th 2008 to June 27th 2008.

Acknowledgments

I would like to thank my supervisor Associate Professor Ekkart Kindler for guiding and helping me through my bachelor project.
Also many thanks to my friend and former classmate Erling Madsen for exchanging information and keeping company to the late hours during the final period of the project.

Kgs. Lyngby, June 2008

Lintalo Suehiro

Contents

Abstract **i**

Preface **iii**

1 Introduction **1**

1.1 The CASE Tool - Section Not corrected yet 2

1.2 Structure of this report 4

2 Brief Overview of the project **5**

2.1 Name, date and place 5

2.2 Current situation 5

2.3 Need 6

2.4 Scope 6

2.5 Span 6

2.6 Synopsis 6

2.7 Assumptions and dependencies 6

3 Analysis **7**

3.1 The project given in the course of SE2 7

3.2 Understanding the current project 8

3.2.1 Finding the problems 8

3.2.2 Known problems with the tools 9

4 Used Technology **13**

4.1 Eclipse 14

4.2 Eclipse Modeling Framework 15

4.2.1 EMF Projects 15

4.2.2 EMF Models 15

4.2.2.1 Ecore model 15

4.2.2.2 Gen Model 15

4.2.3 Generation of projects 16

4.3 Graphical Modeling Framework 17

4.3.1 GMF Models 17

4.3.2 Generation of projects 18

5 implementation **19**

5.1 Making the toolset to work 19

5.2 Integration of the projects 20

5.2.1 Step 1: Integration of components definition editor and deployment editor 20

5.2.2 Step 2 : Integration of interaction editor 20

5.2.3 Step 3 : Integration of the dashboard 21

5.2.3.1 Threading problems 22

6 Conclusion **25**

6.1 Problems occured during the project 25

6.2 Unsolved problems 26

6.3 Conclusion 27

A Installation guide **29**

A.1 Clean installation of Eclipse 29

A.2 Installation of Subclipse 33

A.3 Installation from the CD 33

B User guide **37**

C List of Projects **39**

Chapter 1

Introduction

Developing embedded software systems is a complicated matter. There are numerous developing methods to support projects and one of the way of doing that is the MDD [1].

Our software system is intended to be able to model, simulate, and visualizing behaviors of an embedded system, which we call a CASE [2] tool.

For software development the Unified Modeling Language is quite often used. When a software system is going to be designed, we make use of some of the diagrams of UML like class diagrams and sequence diagrams with help of graphical diagramming software like Visio and then manually write the code that corresponds to the diagrams. This process is time consuming as time is needed to write the diagrams and thereafter make the diagrams into working code.

By removing the step of manually writing the code from the UML diagrams, errors would be minimized and time would be saved. This is also the case with development of embedded systems.

We will use MDD for the development of the CASE tool so we have to code less manually. And the model that is made by the CASE tool use MDD, so no

[1] Model Driven Development
[2] Computer Aided Software Engineering

coding is needed to simulate the model.

The technology we will use is EMF - Eclipse Modeling Framework. With EMF we are able to generate Java code from class diagrams, and it is a framework that can be used to easily create a Eclipse-like applications.

1.1 The CASE Tool - Section Not corrected yet

The CASE tool which is the development subject of this bachelor project, deals with complete to make a simulation running.

sequence(interaction) diagrams, deployment diagrams and component diagrams. It has additionally a dashboard, which models a car wiper system that can is visualized on-screen.

Embedded systems consists of different parts, which we call components. We can see in our specific case in figure 1.1 that our system consists of five components; An ignition switch, a rain sensor, a wiper control and two wipers. We can also see that the ignition switch and rain sensors and the wipers are connected to the wiper control. The boxes represents the components and the lines with small boxes represents the ports. The lines represent connectors. The ports are used to send and receive messages.

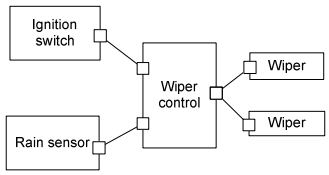


Figure 1.1: Deployment Diagram

With the deployment diagram we see which components that are present and how the components are connected to each other. However if we needs to model the behavior of the components, we would need to use the component definition editor. Here we would define the state machines of the components. The state changes after a received message sent through the ports that connects the components. This is visible in figure 1.2. The state where there is an arrow which points from nowhere, is the initial state. In this figure, **Engine Off** is the initial state. when ignition is turned on or wiper turned off, the current

state is moved to **Wiper off**. The state changes in the direction of the arrows to external event.

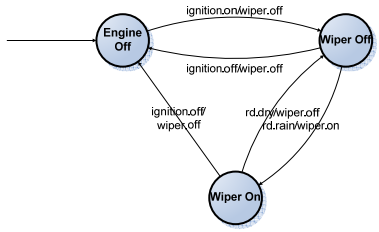


Figure 1.2: Component Definition

With the component editor, we define the behavior of a component, but when we need to get overview of the system we would need the interaction recorder. With the interaction recorder shown in figure 1.3 the interaction of the components can be recorded by a simulation to see if the behavior is correct. If the system behaves in a way that was not intended, we have to go back and change the components definitions and the deployment.

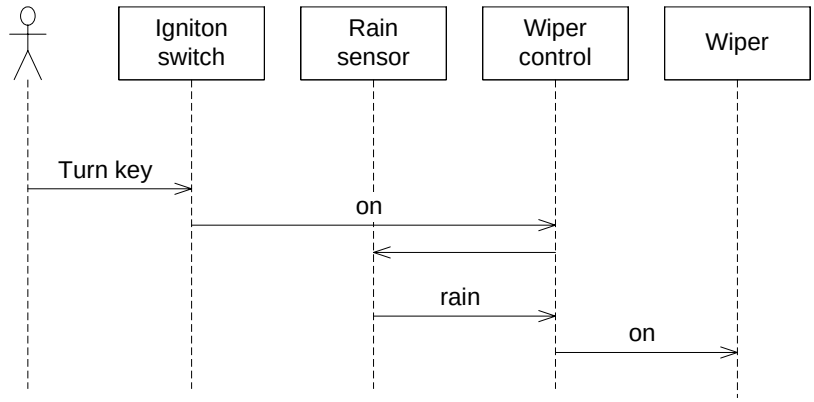


Figure 1.3: Interaction Diagram

To see our modeled system in action, we would need a dashboard.

At the dashboard of our car wiper system in figure 1.4, ignition switch, a wiper and rain are visualized. By simulating the system, the wipers will activate if there is rain and the ignition switch is on.

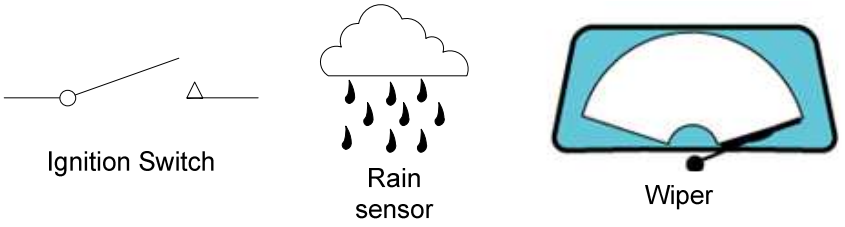


Figure 1.4: Dashboard

1.2 Structure of this report

The overview of the report is as follows.
In chapter 4 we explain how Eclipse, EMF and GMF works.
In chapter 2 the practical information of the project is listed.
Chapter 3 describes the current situation of the progress of the different parts of the project.
Analysis of what is made and what still needs to be done is clarified here.
The implementation of the project in chapter 5 and finally the conclusion of this project.

Chapter 2

Brief Overview of the project

In this chapter we present you a brief overview of the project.

2.1 Name, date and place

The CASE tool project is described and developed by Lintalo Suehiro with guidance of Associate Professor Ekkart Kindler at Technical University of Denmark as a B.Eng. thesis in the period of February 4th to June 30th 2008.

2.2 Current situation

The CASE [1]tool which is intended to model, analyze, simulate and visualize embedded systems, consists currently of four separate systems. The systems contains some unsolved bugs and there are still features that are not implemented. The list of problems is described in section 3.2.2.

[1]Computer Aided Software Engineering

2.3 Need

Integration of the four parts is needed to create the complete CASE tool. The bugs of the systems needs to be fixed after the integration process. The missing features of the systems must be clarified and implemented as time allows.

2.4 Scope

The Scope of this project would be to fully implement all the parts of tool.

2.5 Span

The span would be the subset of the complete model, which deals with deployment, sequence and component diagrams and a dashboard to model an embedded system.

2.6 Synopsis

The project is to develop a CASE tool for embedded systems, which helps the developers by generating code from different UML diagrams.

2.7 Assumptions and dependencies

The technology that is used is Eclipse. For this project three plug ins are used; EMF, GMF and GEF.

Chapter 3

Analysis

In this chapter we will analyse the project description. Let us start with what the project in the course of Software Engineering 2 at Technical University of Denmark was about.

3.1 The project given in the course of SE2

At the course of Software Engineering 2 in the fall semester of 2007, a project description for a CASE tool for modeling, analysing, simulating, and visualizing embedded systems was assigned to the participating students. A presentation on what the tool is going to be capable of was given in the beginning of the course, but at that point there were no clear requirements on the tool. To find out what the requirements of the CASE Tool would be was also a part of the assignment.

The course participants were split up into four groups, as the CASE tool consisted of four different parts and a simulator. The four parts of the CASE tool were as follows.

- The component definition editor, where the automata of the individual

components are defined.

- The deployment editor where the behavior across diffent components were to be defined
- The interaction recorder to see if component definition and the deployment behaves as expected.
- The dashboard to visualize the simulation of the model.

Additionally, each group had to develop a simulator for their part of the tool. The definition of the CASE tool was given with a class diagram which shows how the components are connected to each other. This is visible in figure 3.1. The letters on the figure is very small, so look at this report as a pdf file on the provided CD-ROM.

As this report would act as an overview to the reports made by the groups we do not choose to go into the technical details of the different tools. We refer to the reports of the different groups.

- Group 1 has been working on the deployment diagram editor. [AHH+07]
- Group 2 has been working on the interaction recorder. [ABK+07]
- Group 3 has been working on the dashboard [LJT+07]
- Group 4 has been working on the component definition editor. [AAC+07]

3.2 Understanding the current project

The main topic of the current project is integration of the tools. To integrate the tools, we need to understand the requirements of the individual parts. The tools have different unwanted behavior and it is our task to find them.

3.2.1 Finding the problems

For finding the current problems with the different parts of the tool, we need to follow the use cases written in the report provided by the different groups.
Each part is installed in different workspaces to avoid potential new bug created from the integration.

At the point of making the projects run, we already faced into problems. Due to a classpath file missing, Eclipse could not recognize the structure of the folders for the EMF project. Some other problems was due to missing library files, so most of the projects was unable to run out of the box.

3.2.2 Known problems with the tools

After clearing the errors due to missing files and library files, we have discovered the problems with the tool by trying it out.

- Inconsistency with numbering for the Dashboard, as the different parts get mixed up
- Images of the dashboard is not all visible
- Integration of interaction and dashboard simulator
- Simulator window must be open before running the interaction simulation
- Recorded simulation does not appear before saving and reopening the CASEtool.
- Example in the Dashboard is not complete
- arrow from state of one component over to a state of an another component
- Category needs to be in one place
- Simulator button from deployment editor should be removed
- Load new dashboard diagram would be nice
- Automaton can be removed out of components when moved to left/up, because the component box will not expand. If it is moved so much up or to the left, that automaton box does not overlap with the component box, it is not possible to pick it up later with the mouse.
- Once a state has been chosen as initial, it appears not to be possible to change others to initial. It is updated if we click on the automaton box.
- The transition does not change when the source or target is changed in the property window
- Graphical repair of initial state. The name of the first created state is changed to →a, if the name of the state was “a”. In the property window, there is still a state named “a”, even though there are no states named “a” anymore on the editor.

- If you select a software component and put an automaton box in it, then put states in it, the states are labeled false, which is not the case with sensors and actuator.
- problems with state names which length is 1 like “a”,”b” to open with tree editor

Due to the many obstacles that appeared during the project, where some were hard to solve due to lack of documentation, most of them is still unsolved.

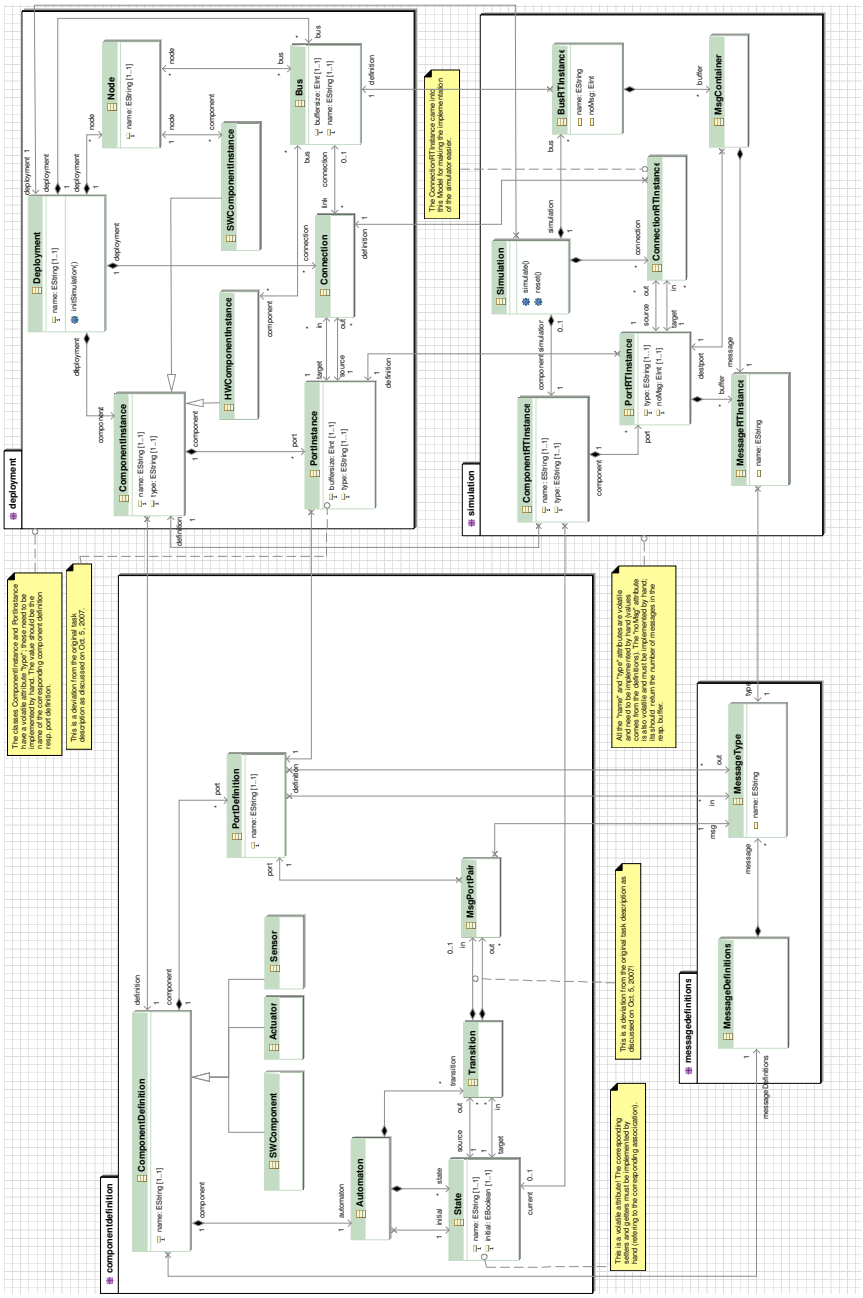


Figure 3.1: Definition of CASETool

Chapter 4

Used Technology

To be able to understand how we dealt with this project, we need to introduce the technology that is used in this project. We use a IDE called Eclipse, one of the most used environment for use with software development with Java. The main reason we chose to use Eclipse is the model driven development capability of Eclipse Modeling Framework.

This chapter is an introduction to Eclipse and EMF, so by reading this we will be ready to read the next chapter where we discuss about the implementation of the integration of the project.

To make the tool graphical we use Graphical Modeling Framework and in addition Graphical Editing Framework. GMF simplifies the steps to make a visual editor compared to GEF, and this is the reason we choose to mainly use this framework. In the next sections we go through the mentioned technologies in details.

4.1 Eclipse

Eclipse is one of the most known IDEs [1] for development with Java. The facilities of eclipse reminds very much of Netbeans [2], so there are many convinces for the developer like overview of the projects, classes, methods and variables, syntax highlighting, searching facilities, debugging and many more. A screen of eclipse is shown in figure 4.1

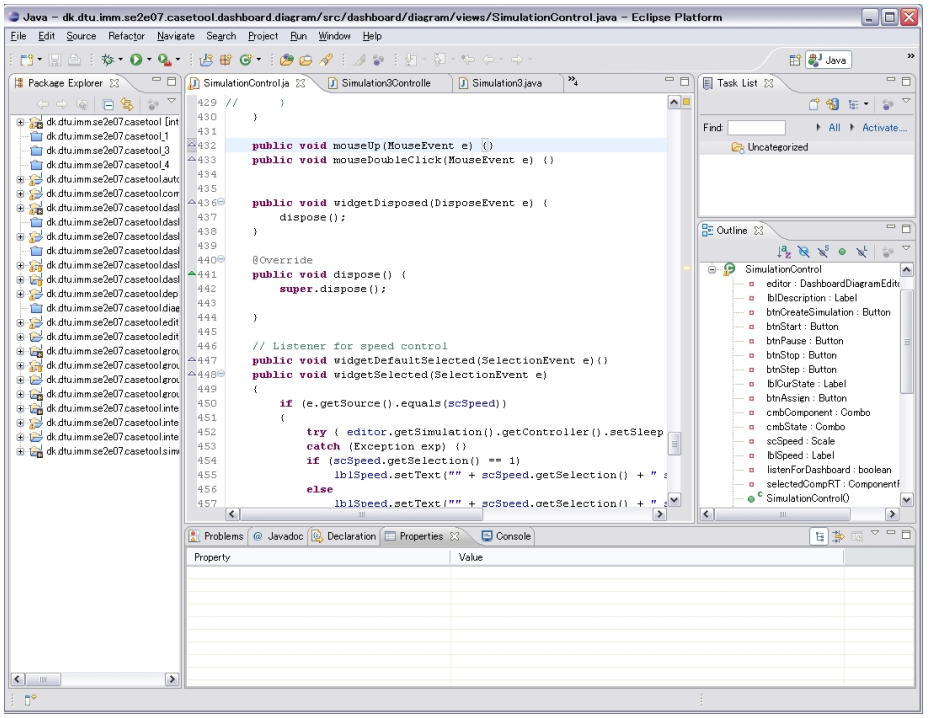


Figure 4.1: Screen of Eclipse

Eclipse has a plug-in system, so there exist many kinds of plug-in you can use this IDE for; not necessarily only for Java, for example there exist plug-ins for software development with C or C++.

As explained in the beginning of this chapter, the main reason we use Eclipse is the because of the model driven development facilities of Eclipse Modeling Framework. The details of EMF is explained in the next section.

[1] Integrated Development Environment
[2] an IDE supported by Sun Microsystems

4.2 Eclipse Modeling Framework

This section is a guide for what different EMF projects in Eclipse do and how they are different from standard Java projects. To be able to navigate around the different projects and understand what they do, we need to introduce the few elements of EMF.
We will start by explaining what EMF projects contains.

4.2.1 EMF Projects

EMF projects contains more different files and folders compared to standard Java projects in Eclipse. Where standard Java projects have a source folder and a Source Library, The EMF projects contains additionally Plug-in Dependencies which major part is consisted of eclipse relevant library files, a few configuration files and a model folder where we find Ecore models and Gen models. Ecore models and Gen models is explained in the next subsection.

To create an EMF project, we firstly need a ecore model file which is required under the wizard of creating a new EMF project. A genmodel is there after created.

4.2.2 EMF Models

4.2.2.1 Ecore model

To make a model in EMF, we need a ecore model to describe how a model is constructed. The ecore model is similar to class diagrams where we describe how each class is associated to an another. Eclipse provides a default tree editor to write the ecore model. You can also use the ecore diagram editor which is more convenient to use, as the relations between the objects are represented visually. In figure 4.2 we can see how a Ecore file looks like. A graphical representation of the ecore model will be briefly presented in the chapter 3.

4.2.2.2 Gen Model

The role of the genmodel is to auto generate code from the ecore model we have specified. From the editor of a Genmodel what we see is exactly same as the

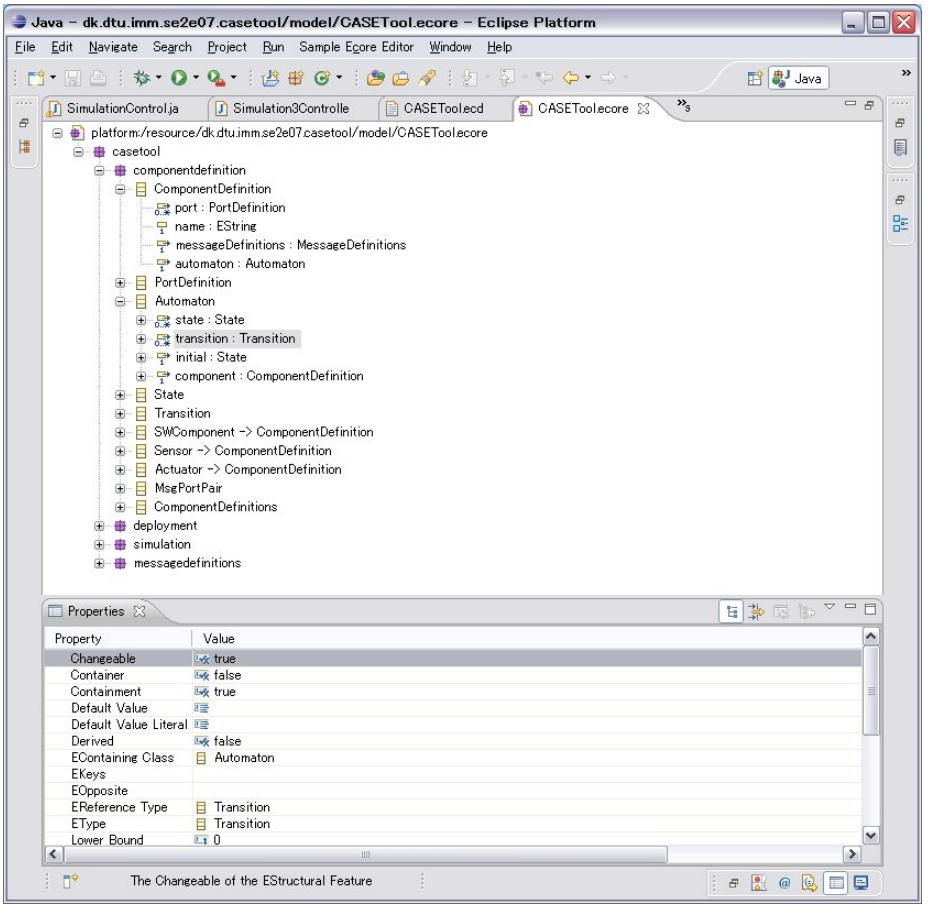


Figure 4.2: Ecore editor

ecore model. However by right clicking on the model, we now have the option to generate model, edit, editor code and test code.

4.2.3 Generation of projects

When we generate the model code, all of the files, including the source folder in the working project now represent the definition in the ecore model. When we by default generate the edit and editor code, new projects are added to the current workspace which is named **.edit** and **.editor** respectively. The editor code is actually not necessary in this projects, as none of the editor need the

XML tree editor. The project would be visible under the Package Explorer. It is shown on figure 4.3.

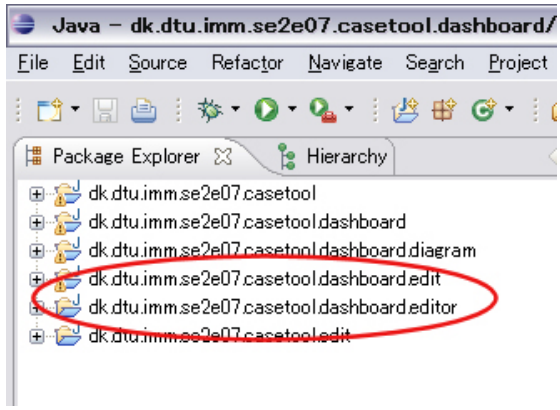


Figure 4.3: Edit and Editor project generated by **genmodel**

To have a graphical editor we need to use Graphical Modeling Framework, which is explained in the next section.

4.3 Graphical Modeling Framework

To make an editor made with our EMF project, we need to make it graphical by using GMF.

There are many settings in the GMF models we can adjust so the editor ,which we are developing, is able to behave in a certain way.

4.3.1 GMF Models

The default folder to locate GMF models is in the same place we locate the EMF models. To be able to generate a diagram model from GMF models, we need to define the **gmfgraph**, **gmftool**, **gmfmap** and **gmfgen**.

gmfgraph is used to defining how figures in the graphical editor looks like, and we use **gmftool** for defining what kind of tool we need, naming of the figures and a like. The **gmfmap** links the figures and tools, which were defined by

gmfgraph and **gmftool**, with the EMF model. When this is done, **gmfgen** can be created.

4.3.2 Generation of projects

Just like genmodels, we can generate code by right clicking on the models. By choosing generate diagram code, a new project with **.diagram** is auto generated. The project would be visible under the Package Explorer. A screen capture of it is visible on figure 4.4.

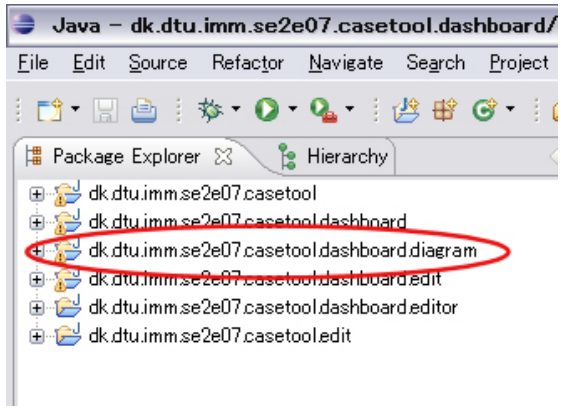


Figure 4.4: Diagram project generated by **gmfgen**

We have briefly gone through the EMF and GMF, and we now understand which projects contain EMF models and which projects are auto generated. We have a We are now prepared to go to the next chapter where we present the implementation part.

Chapter 5

implementation

[BSM+07]

5.1 Making the tool set to work

As mentioned in the previous chapter, none of the projects was unable to run out of the box.
This is due to a bug in subclipse, so not all the files in a project was submitted. For Group 1, the Deployment Editor, junit library was missing. For the Group 2 the interaction recorder, **.classpath** file was missing, so Eclipse could not see the structure of project. This was an error which would be hard to discover, when you just started to use Eclipse and EMF. For the group 3 the dashboard group, ecore.xmi and junit was missing. Regarding group 4, there were problems with the TransitionItemProvider, which we never solved. This is due to the the redefinition of the ecore model, where InOutMessages was removed by the most of the groups.
As no easy solution was found, the code which caused syntax errors was lined out.

After the tool were running, we were looking for problems of the tools. After

the problems were found, integration of the tools started.

5.2 Integration of the projects

The common project of the CASE tool is **dk.dtu.imm.se2e07.casetool**, and there are small differences with the versions each group used. We chose to use of main case tool project of group 2, as it was the closest to the original project. The main project of Group 4, had an small piece of code, which added an → to the component name, so the code was added to our main project.

5.2.1 Step 1: Integration of components definition editor and deployment editor

The integration of components definition editor and deployment editor went without major obstacles. The **dk.dtu.imm.se2e07.casetool** of from the both projects was carefully compared to adapt for both projects to be working simultaneously.

5.2.2 Step 2 : Integration of interaction editor

The integration of interaction editor and deployment editor went also without major obstacles. The **dk.dtu.imm.se2e07.casetool** was compared to the original to adapt for both projects to be working simultaneously.

When we want to create a new diagram on the running application, we click **file→new→Other**. The problem was that each groups had different place to store the diagram file creation. By default it is put under the Examples folder. We found out how to move the diagram file creation of the different folders, but this is by manually editing the plugin.xml file, so when we make changes to the GMF file, the settings would be lost. We found no solution to this problem as there were no sufficient documentation about this.

5.2.3 Step 3 : Integration of the dashboard

The final step of the integration was the dashboard. There were several problems we encountered, while we were integrating the dashboard. As we made changes to the GMFgen file, and auto generated code again, all the figures was wrongly mapped. This was due to the numbering of the GMFgen of the dashboard was not the same; i.e. not the latest one. The numbering was edited back, so the Dashboard figures is correctly mapped again. This is done in the gmfgen 5.1.

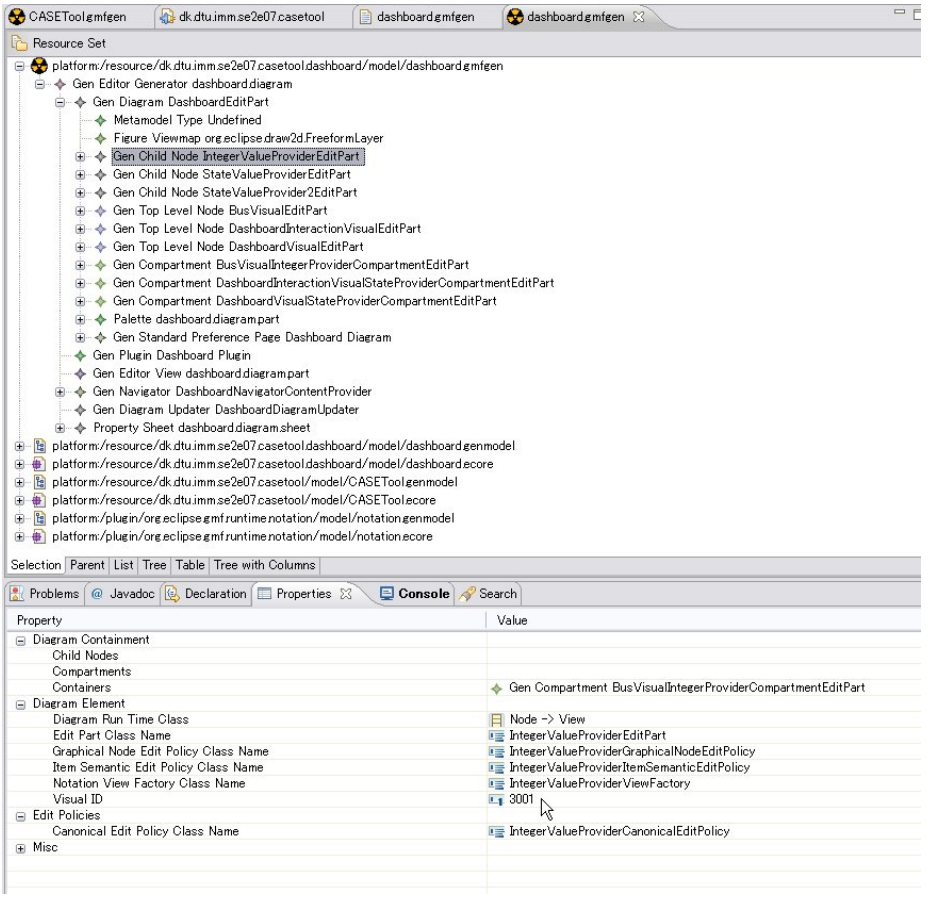


Figure 5.1: Wrong numbering

Even after integrating the dashboard some of the picture were not visible. This was actually caused by the little code, which added a → to the component

names, and therefore the Dashboard was not able to locate the correct picture.

5.2.3.1 Threading problems

Eclipse does not allow multiple threads to modify the same model, as the main thread of eclipse has the exclusive rights. Due to this problem, the interaction diagram is not working correctly and the dashboard is throwing an interrupted exception which is not visible to the user.

By using methods of the Display class, it is possible to make the main thread of Eclipse to execute code. By doing this, changing rights between threads is not neccesary anymore and the problems of interrupting the simulation thread would be excess. We have applied this to the simulator of the dashboard; Simulator3Thread and Simulator3Controller. The Simulator3Thread is now not threaded anymore and the play method of Simulator3Controller call the Display thread to execute a same code as original simulator code. Originally, The simulator threw a Interruptexception when the simulation was stopped, but now it simply exits the thread. When play method is called again, the Display thread is once again called. The code of the classes can be seen on figure 5.2.

Unfortunately, because of lack of time, the problems with the interaction recorder was not fixed.

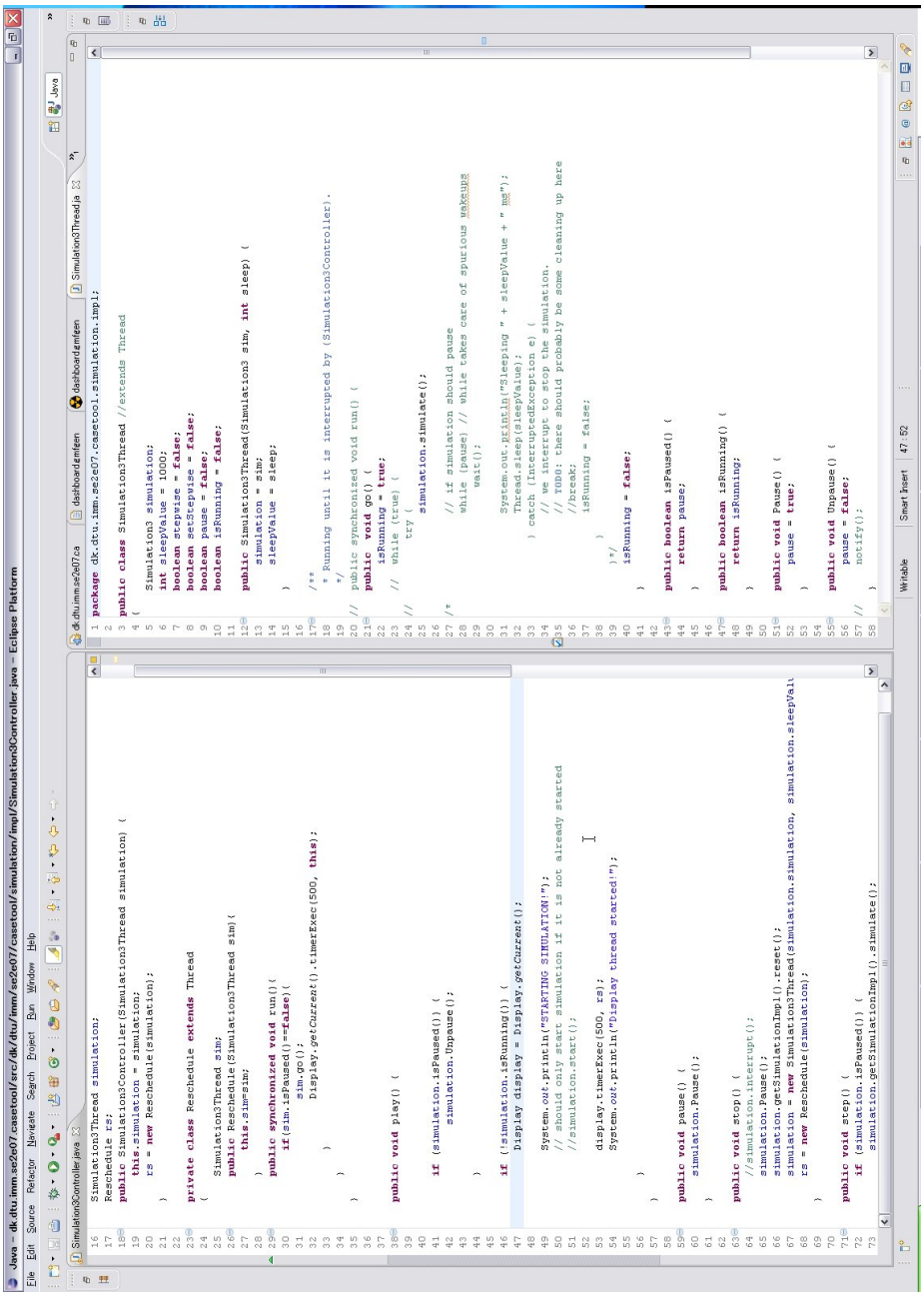


Figure 5.2: Simulator code

Chapter 6

Conclusion

We would like to state the problems which occurred during the project, list up the problems which still needs to be resolved and at last conclude this thesis in this chapter.

6.1 Problems occurred during the project

There were numerous problems and difficulties we faced during the development of the project.

The first thing was to understand what the project was about and how much the individual groups had progressed with their tool during the Software Engineering 2 course in the fall semester of 2007.

The next step was understand how to use Eclipse and and how EMF and GMF works.

The existing tutorials and documents regarding how to get started was difficult to understand and there existed no other easily obtainable alternatives.
The limited documentation on the EMF and GMF was also an problem for the practical use of the technology. Many hours was spent for looking on the net to

how to solve a specific problem, which in it self was not a complex problem, but since there were no documentation for how to do it, there were nothing to do.

For the installation of Eclipse, we faced some difficulties as there were no instructions to install which packages to make the projects work. The installation process was not documented in the reports of the different groups, so it would seem that the installation would be a trivial task, but it was rather winding.
Installing the EMF and GMF packages to Eclipse to make the projects work would seems to be the natural step, however this was not the case.

Due to some of the files for the project of the groups was missing, the project did not work out of the box.
This was due to bugs in subclipse, so not all files was submitted to the svn server.

Because there were missing packages to the projects we checked out, a we manually added the missing library files.
As we were used to Netbeans, we looked at the properties of the projects and added the missing library files to the build path.

This made the error notifications on the code due to missing library files disappear and it seemed that the projects was working fine. However this led to the running application behave very strange. The application was running but after after a line in the code where the library to use the main thread of Eclipse, the Display thread, was called, there were no obvious reason why it did not work, as no exception was thrown to the console. But

Another major obstacle was that the accessible GMF models of the groups on the svn server, was not the latest versions.
When there were needs to change the GMF models, we had to carefully compare the auto generated code with the code from the originally submitted project and there after make changes to the GMF generation file again. This was also a very time consuming and a frustrating task, as there were myriads of code we had to look through.

6.2 Unsolved problems

The unsolved problems is as follows:

- Integration of interaction and dashboard simulator

- Simulator window must be open before running the interaction simulation
- Recorded simulation does not appear before saving and reopening the CASEtool.
- Example in the Dashboard is not complete
- arrow from state of one component over to a state of an another component
- Category needs to be in one place
- Simulator button from deployment editor should be removed
- Load new dashboard diagram would be nice
- Automaton can be removed out of components when moved to left/up, because the component box will not expand. If it is moved so much up or to the left, that automaton box does not overlap with the component box, it is not possible to pick it up later with the mouse.
- Once a state has been chosen as initial, it appears not to be possible to change others to initial. It is updated if we click on the automaton box.
- The transition does not change when the source or target is changed in the property window
- Graphical repair of initial state. The name of the first created state is changed to →a, if the name of the state was "a". In the property window, there is still a state named "a", even though there are no states named "a" anymore on the editor.
- If you select a software component and put an automaton box in it, then put states in it, the states are labeled false, which is not the case with sensors and actuator.
- problems with state names which length is 1 like "a","b" to open with tree editor

6.3 Conclusion

The different parts of the project is now integrated into a single Tool, so we do not have the need to switch between four instances of eclipse to run our application.

From the point of view of integration of the different parts of the CASE Tool, the minimal requirements was met, as the CASE tool is now integrated into a

single tool. The threading problem of dashboard simulator is also solved, but the interaction recorder problems still remains. However due to many obstacles and difficulties, the progress of the project did not advance as wanted. There are still many improvements to be made, but major obstacles we met which were not written of the previous reports, is now clarified and this would ease up to the next developers that takes over the project.

Appendix A

Installation guide

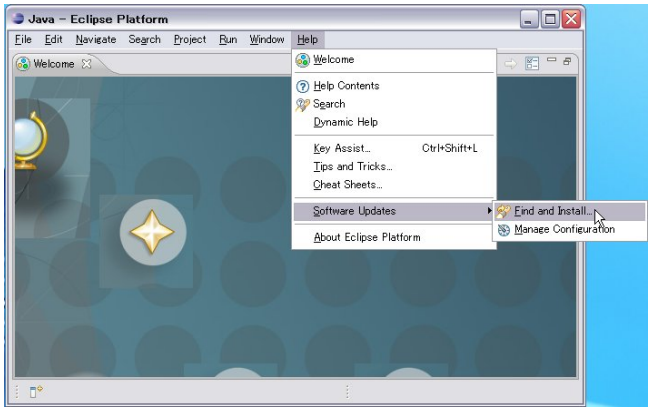
A.1 Clean installation of Eclipse

This chapter instructs how to install eclipse and which package that is needed.

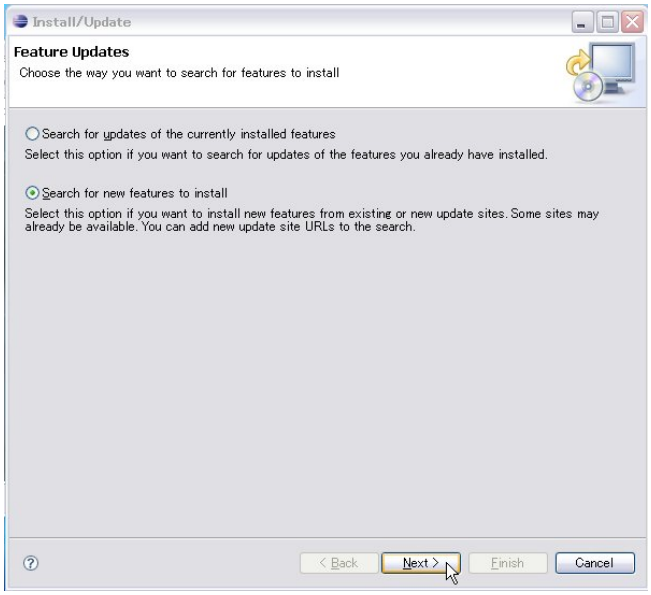
Eclipse is an IDE which is developed constantly, so new versions come out frequently. The Eclipse version that was used in this project was Eclipse 3.2 Europa. There are several versions that are available for download, and the one we used was the java version.

Please remember to install Java SDK and JRE prior to Eclipse.

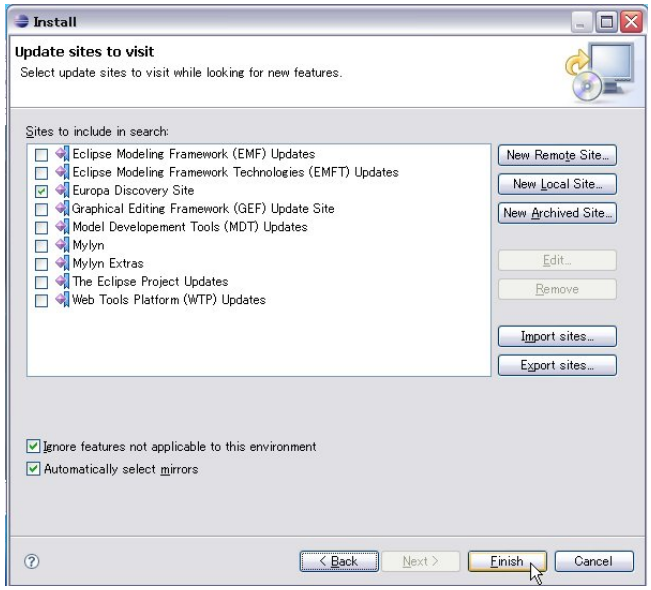
To make GMF and GEF work with Eclipse, we would need to run Eclipse and under help menu, choose Software Updates, find and install.



Choose **search for new features to install** and press **next**.

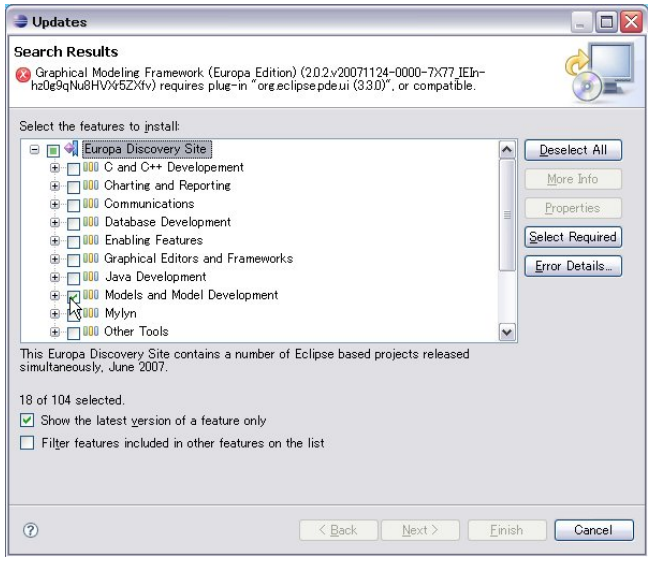

Install/Update
Feature Updates
Choose the way you want to search for features to install
Search for updates of the currently installed features
Select this option if you want to search for updates of the features you already have installed.
Search for new features to install
Select this option if you want to install new features from existing or new update sites. Some sites may already be available. You can add new update site URLs to the search.
< Back
Next >
Finish
Cancel

Check **Europa Discovery Site** and click **finish**.

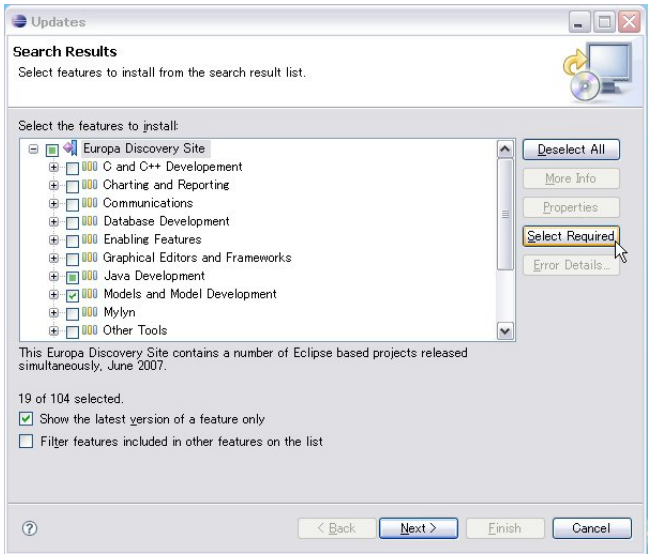


If Eclipse ask for choosing a mirror, check A**automatically select mirrors** and click **ok**.

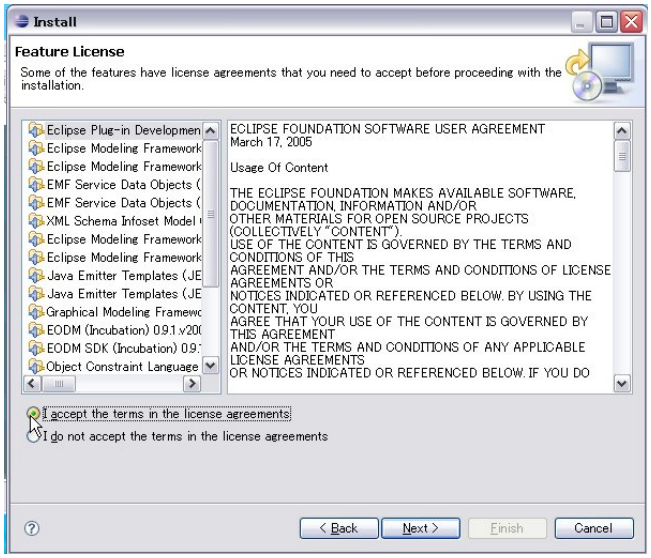
Expand Europa Discovery Site and check **Model and Model Development**.



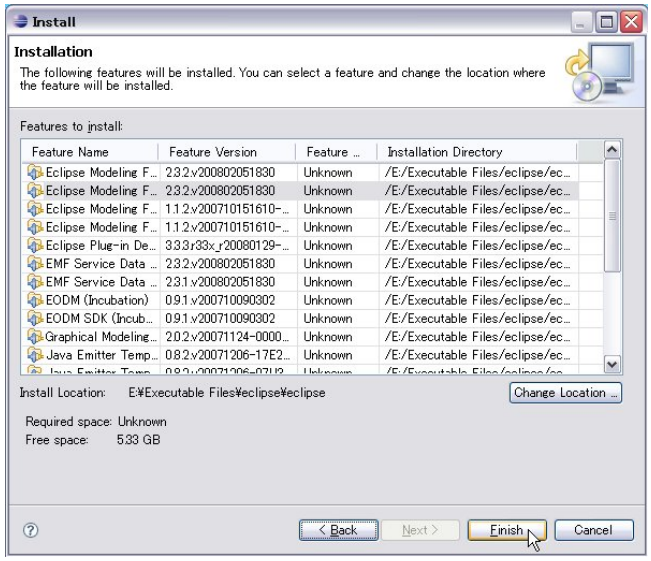
A warning comes out and click on **Select Required** and then **next**.



Check **I accept the terms in the license agreements** and click on **next**.


Install
Feature License
Some of the features have license agreements that you need to accept before proceeding with the installation.
Eclipse Plug-in Developmen
Eclipse Modeling Framework
Eclipse Modeling Framework
EMF Service Data Objects (
EMF Service Data Objects (
XML Schema Infoset Model
Eclipse Modeling Framework
Eclipse Modeling Framework
Java Emitter Templates (JE
Java Emitter Templates (JE
Graphical Modeling Framewo
EODM (Incubation) 0.9.1 v20
EODM SDK (Incubation) 0.9.
Object Constraint Language
ECLIPSE FOUNDATION SOFTWARE USER AGREEMENT
March 17, 2005
Usage Of Content
THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO
I accept the terms in the license agreements
I do not accept the terms in the license agreements
< Back
Next >
Finish
Cancel

Click **Finish**.


Install
Installation
The following features will be installed. You can select a feature and change the location where the feature will be installed.
Features to install:
Feature Name
Feature Version
Feature ...
Installation Directory
Eclipse Modeling F... 2.3.2.v200802051830 Unknown /E:/Executable Files/eclipse/ec...
Eclipse Modeling F... 2.3.2.v200802051830 Unknown /E:/Executable Files/eclipse/ec...
Eclipse Modeling F... 1.1.2.v200710151610-... Unknown /E:/Executable Files/eclipse/ec...
Eclipse Modeling F... 1.1.2.v200710151610-... Unknown /E:/Executable Files/eclipse/ec...
Eclipse Plug-in De... 3.3.3r33x_r20080129-... Unknown /E:/Executable Files/eclipse/ec...
EMF Service Data ... 2.3.2.v200802051830 Unknown /E:/Executable Files/eclipse/ec...
EMF Service Data ... 2.3.1.v200802051830 Unknown /E:/Executable Files/eclipse/ec...
EODM (Incubation) 0.9.1.v200710090302 Unknown /E:/Executable Files/eclipse/ec...
EODM SDK (Incub... 0.9.1.v200710090302 Unknown /E:/Executable Files/eclipse/ec...
Graphical Modeling... 2.0.2.v20071124-0000... Unknown /E:/Executable Files/eclipse/ec...
Java Emitter Temp... 0.8.2.v20071206-17E2... Unknown /E:/Executable Files/eclipse/ec...
Install Location: E:¥Executable Files¥eclipse¥eclipse
Change Location ...
Required space: Unknown
Free space: 5.33 GB
< Back
Next >
Finish
Cancel

Eclipse is now ready to use the GMF and GEF, we only need one thing to be able to use the tool.

A.2 Installation of Subclipse

We need to install the sublipse plugin to be able to check out the projects from the SVN server. There is an intuitive installation guide on the website of Subclipse. Go to *http://subclipse.tigris.org/install.html to be guided on the installation.

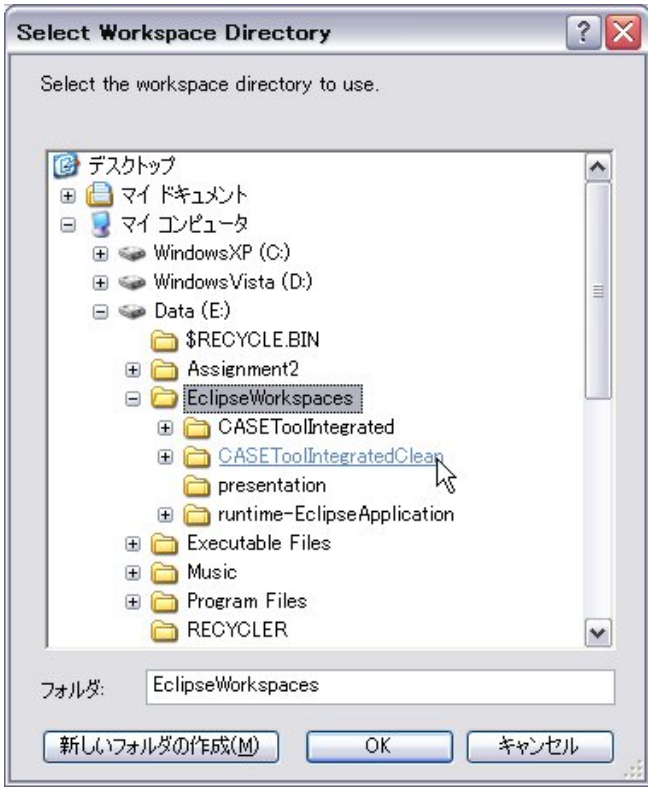
All the plug ins are installed now, and we now need to connect to the SVN server. For security reasons of the SVN server, we do not show the ip address on this thesis.

After connecting to the server with subclipse, you are now ready to use Eclipse.

A.3 Installation from the CD

The easiest way to run the Eclipse is installing from the provided CD. There are actually no installation processes, you just move the files from the CD to a

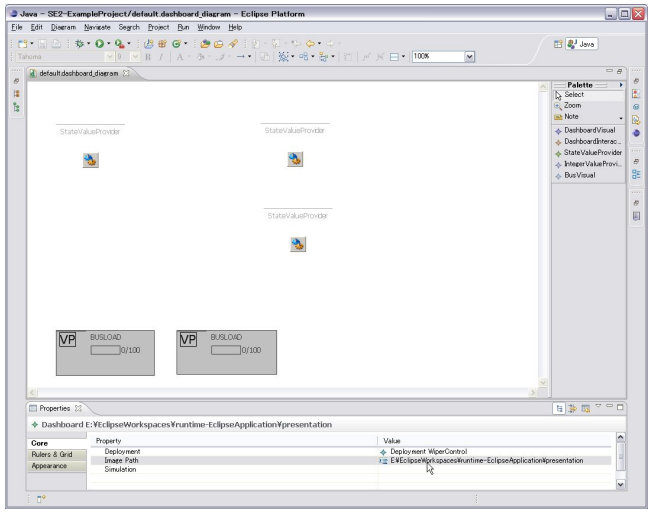
local folder.
Remember to install java prior to running Eclipse, as otherwise it would not run.
At the root directory of the CD Rom drive, there are three folders: Eclipse, EclipseWorkspaces and BEngThesis.
Move Eclipse and EclipseWorkspaces to a local folder of your choice.
Start Eclipse up by double clicking on it and select the path you have moved the EclipseWorkspaces onto.
Choose EclipseWorkspaces\CASEToolIntegratedClean which contains only the new projects or
EclipseWorkspaces\CASEToolIntegrated which contains all of the projects from the SVN server and the new projects.



Click **OK** and **OK** again.

Eclipse should now run and load the projects.

In the EclipseWorkspaces folder, a folder named presentation is there. This folder contains all the icons for the dashboard. On the dashboard window, right click to open the property window. Paste the path where you have moved the presentation to.



Appendix B

User guide

As this thesis is

For the user guide of the interaction editor, please look at [ABK+07], the user manual of the component editor please look at [AAC+07], the user manual of the deployment editor please look at [AHH+07] and for the user guide for the please look at [LJT+07].

Appendix C

List of Projects

A overview of the projects is shown in this chapter. There are following projects in the CASE Tool:

- dk.imm.se2e07.casetool
- dk.imm.se2e07.casetool_1
- dk.imm.se2e07.casetool_2
- dk.imm.se2e07.casetool_3
- dk.imm.se2e07.casetool.components.diagram
- dk.imm.se2e07.casetool.dashboard
- dk.imm.se2e07.casetool.dashboard.diagram
- dk.imm.se2e07.casetool.deployment.diagram
- dk.imm.se2e07.casetool.edit
- dk.imm.se2e07.casetool.editor
- dk.imm.se2e07.casetool.interaction

- dk.imm.se2e07.casetool.interaction.diagram
- dk.imm.se2e07.casetool.interaction.edit
- dk.imm.se2e07.casetool.simulation

We will explain the projects one by one.

dk.imm.se2e07.casetool

This is the main project which contains the emf model. The other projects depend on the source code in this project and therefore when the source code is edited, the change would affect all on the other projects. As Group 2 has choosed to use dk.imm.se2e07.casetool.interaction to contain the GMF generator file, the GMF generator file of this project is not used.

dk.imm.se2e07.casetool_1

This project contains GMF model of deployment editor and the GMFgen of this projects creates the deployment.diagram. It is used to make changes in the deployment.diagram and compare the differences with the main project.

dk.imm.se2e07.casetool_3

This project is used to make changes in the deployment.diagram and compare the differences with the main project. Group 3 has made a separate GMF model which is located in dk.imm.se2e07.casetool.dashboard.

dk.imm.se2e07.casetool_4

This project contains GMF model of components editor and the GMFgen of this projects creates the components.diagram. It is used to make changes in the deployment.diagram and compare the differences with the main project.

dk.imm.se2e07.casetool.components.diagram

This project was auto generated from GMFGen of dk.imm.se2e07.casetool_4. It contains code regarding the graphical parts of the component definition editor.

dk.imm.se2e07.casetool.dashboard

This project contains the GMF model for the dashboard and dashboard.diagram will be created from this project.

dk.imm.se2e07.casetool.dashboard.diagram

This project contains code regarding the graphical parts of the dashboard.

dk.imm.se2e07.casetool.deployment.diagram

This project contains code regarding the graphical parts of the deployment editor.

dk.imm.se2e07.casetool.edit

This project was auto generated by the genmodel of the main project. The project is needed to be able to edit on the editors.

dk.imm.se2e07.casetool.interaction

This project contains the GMF model for the interaction recorder and interaction.diagram will be created from this project.

dk.imm.se2e07.casetool.interaction.diagram

This project contains code regarding the graphical parts of the interaction editor.

dk.imm.se2e07.casetool.interaction.edit

This project was auto generated by the genmodel of the interaction recorder. The project is needed to be able to edit on the interaction recorder.

dk.imm.se2e07.casetool.simulation

This project contains the simulator code for the interaction recorder.

Bibliography

[AAC+07] K. Ahrensberg, O. Ahtirschi, L. Christensen, P. T. Christensen, E. Gökten, X. Moreels, and S. Valberg. Software requirement specification - case tool application: Components definition editor and simulation. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, 2007.

[ABK+07] A. Amini, P. M. Back, J. Kristensen, H. Mynderup, S. H. Pedersen, and D. Schiemer. System specification case tool for developing embedded systems - with main focus on interaction editor/recorder. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, 2007.

[AHH+07] M. H. Andersen, J. Hansen, B. P. Hansen, M. A. Müller, A. Pedersen, M. Sørensen, T. L. Vestergaard, and P. Wind. System specification - revised. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, 2007.

[BSM+07] Frank Budinsky, David Steinberg, Ed Merks, Raymond Ellersick, and Timothy J. Grose. *Eclipse Modeling Framework*. Addison Wesley, 2007.

[LJT+07] D. V. Lebech, A. S. Jensen, M. Thomassen, M. Andersen, K. Andersen, M. D. Anyaogu, and R. O Galve. Case tool editor - dashboard documentation. Technical report, Informatics and Mathematical Modeling, Technical University of Denmark, 2007.